Expose ports

There are a few ways to expose ports on your pod to the outside world. The first thing that you should understand is that the publicly exposed port is most likely NOT going to be the same as the port that you expose on your container. Let's look at an example to illustrate this.

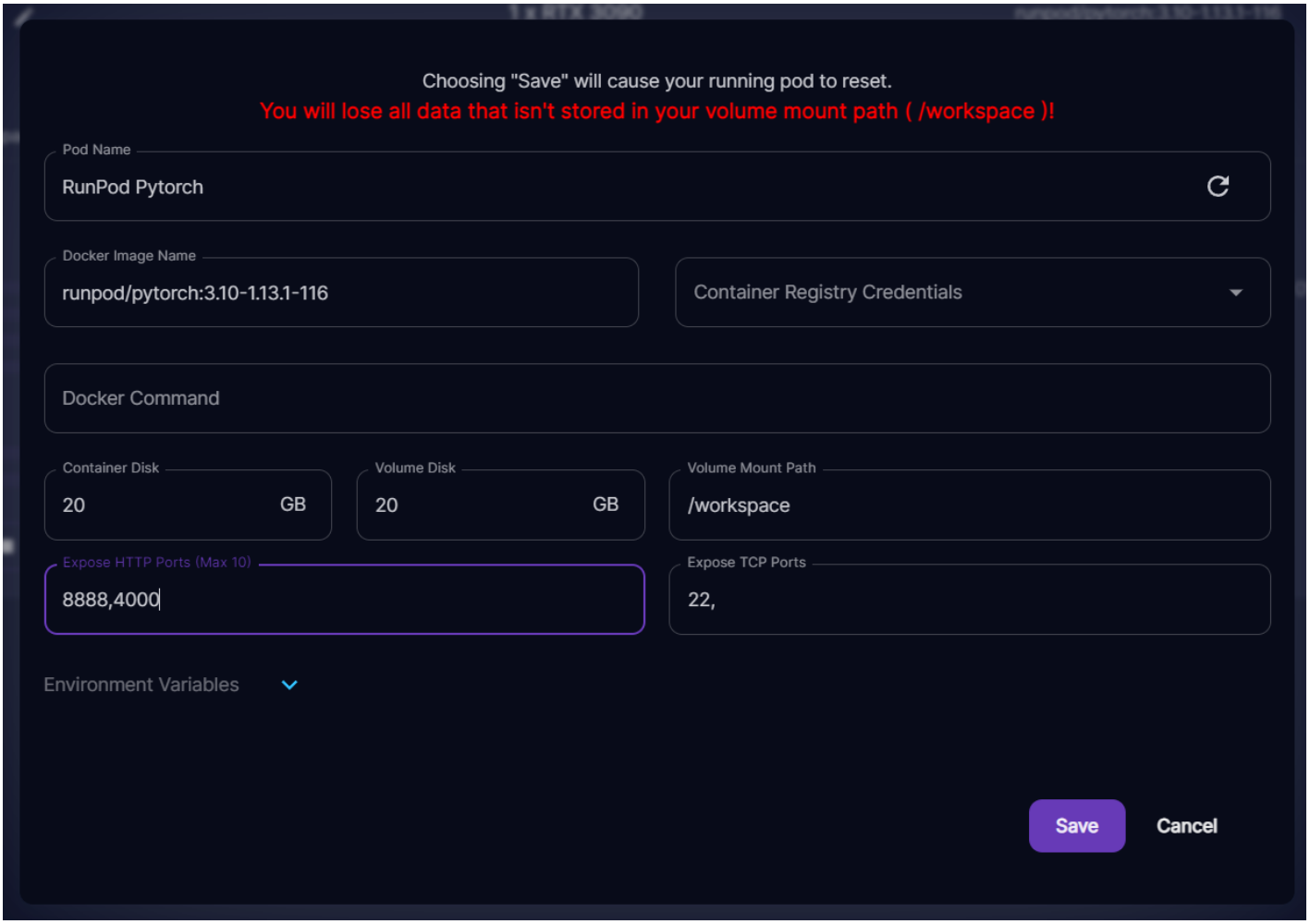
Let's say that I want to run a public API on my pod using uvicorn with the following command:

```
uvicorn main:app --host 0.0.0.0 --port 4000
```

This means that uvicorn would be listening on all interfaces on port 4000. Let's now expose this port to the public internet using two different methods.

Through RunPod's Proxy

In this case, you would want to make sure that the port you want to expose (4000 in this case) is set on the Template or Pod configuration page. You can see here that I have added 4000 to the HTTP port list in my pod config. You can also do this on your template definition.



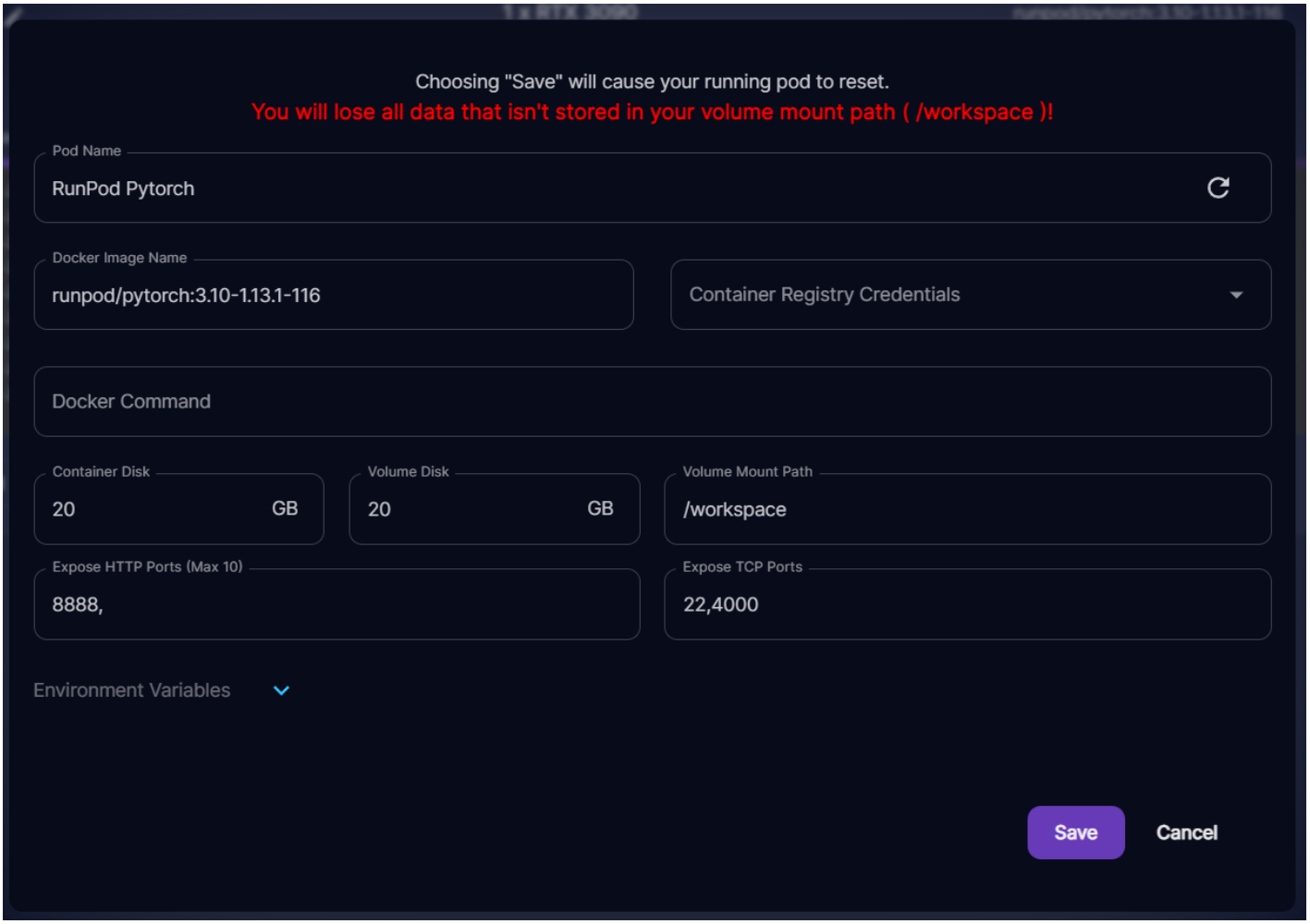
Once you have done this, and your server is running, you should be able to hit your server using the pod's proxy address, which is formed in this programmatic way, where the pod ID is the unique ID of your pod, and the internal port in this case is 4000:

```
https://{POD_ID}-{INTERNAL_PORT}.proxy.runpod.net
```

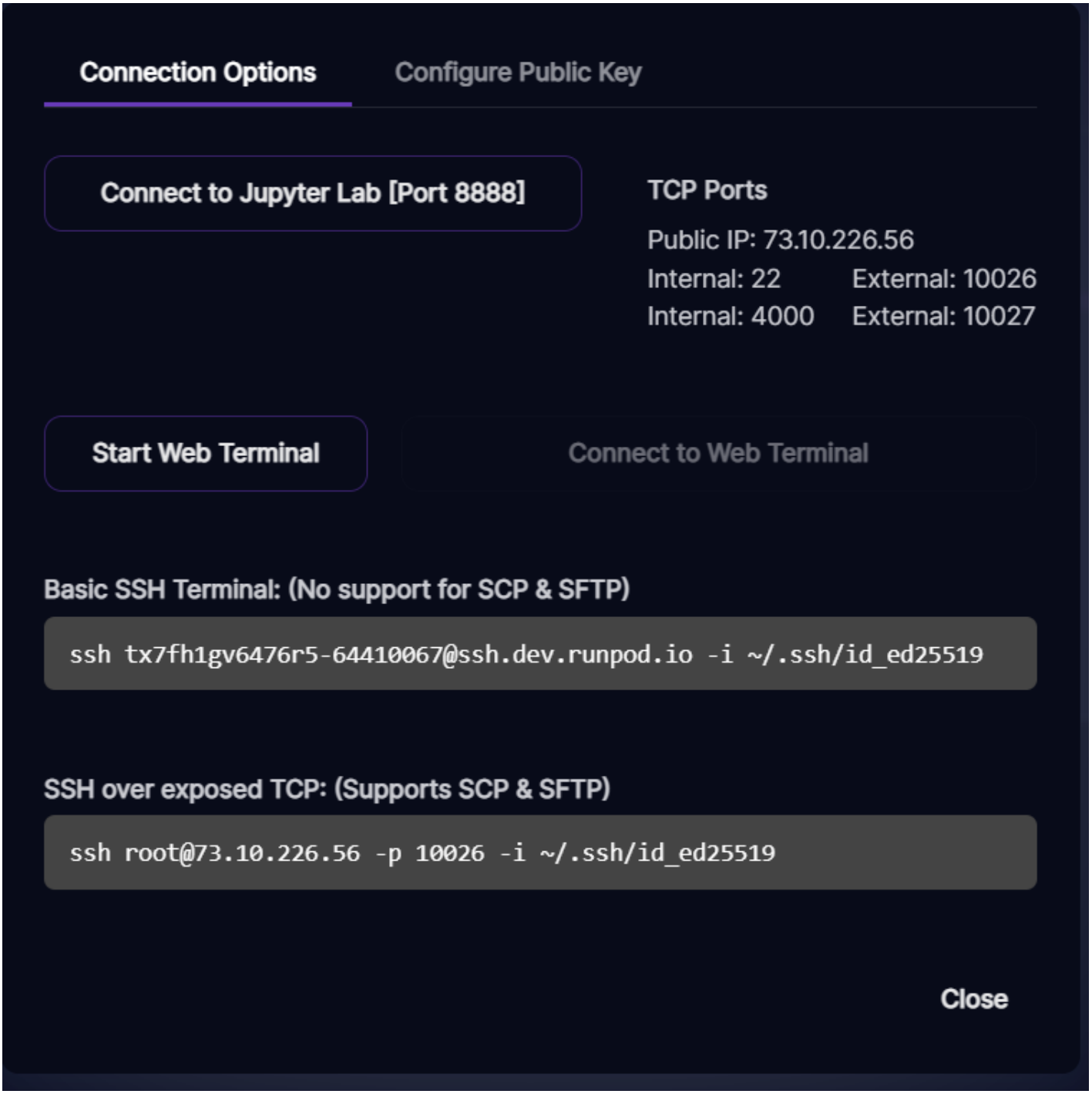
Keep in mind that this exposed to the public internet. While your pod ID can act as a password of sorts, it's not a replacement for real authentication, which should be implemented at your API level.

Through TCP Public IP

If your pod supports a public IP address, you can also expose your API over public TCP. In this case, you would add the port to the TCP side of the configuration.



The only difference here is that you will receive an external port mapping and a public IP address to access your service. For example, your connect menu may look something like this:



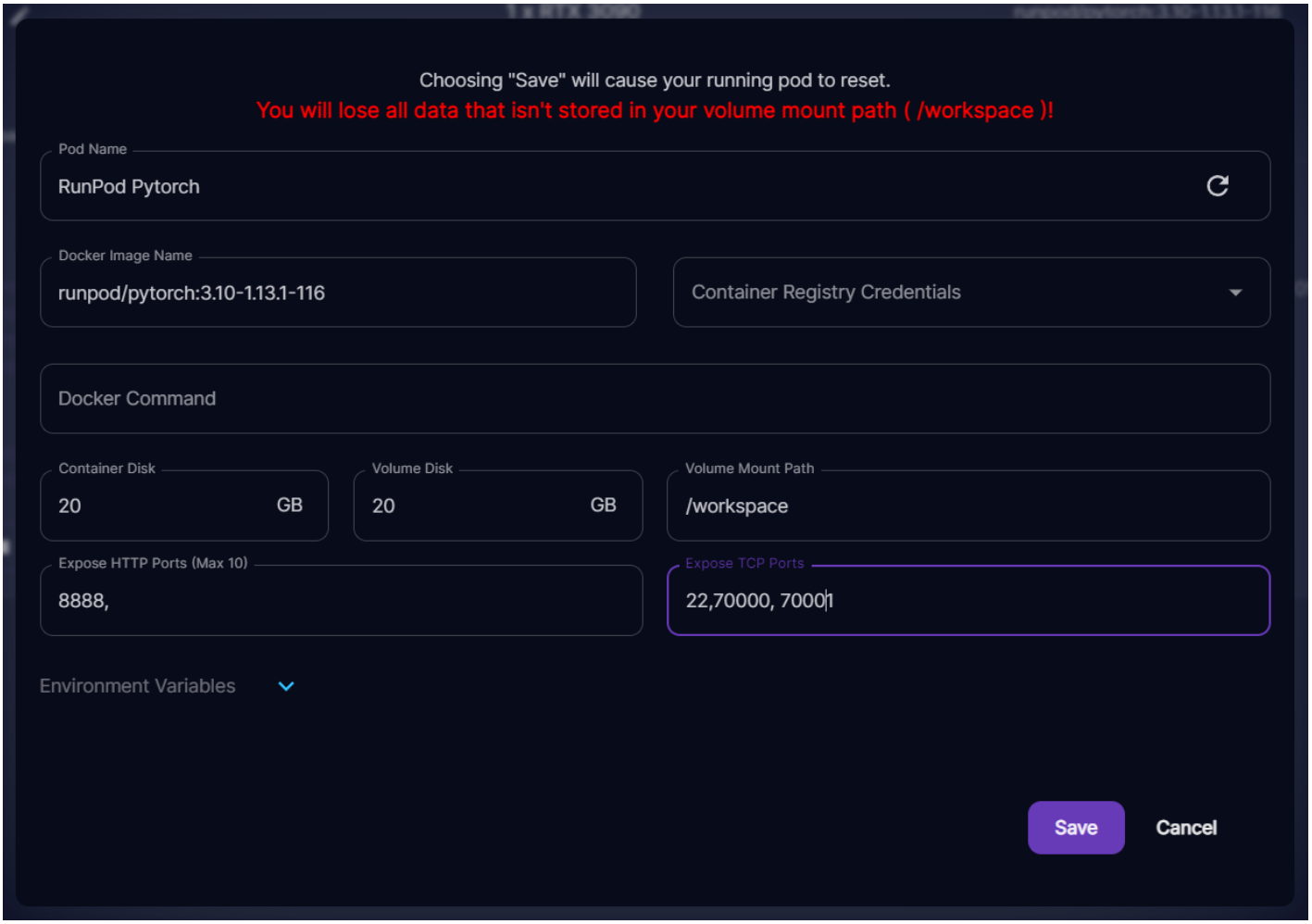
In this case, you would be hitting your service running on 4000 with the following ip:port combination

```
73.10.226.56:10027
```

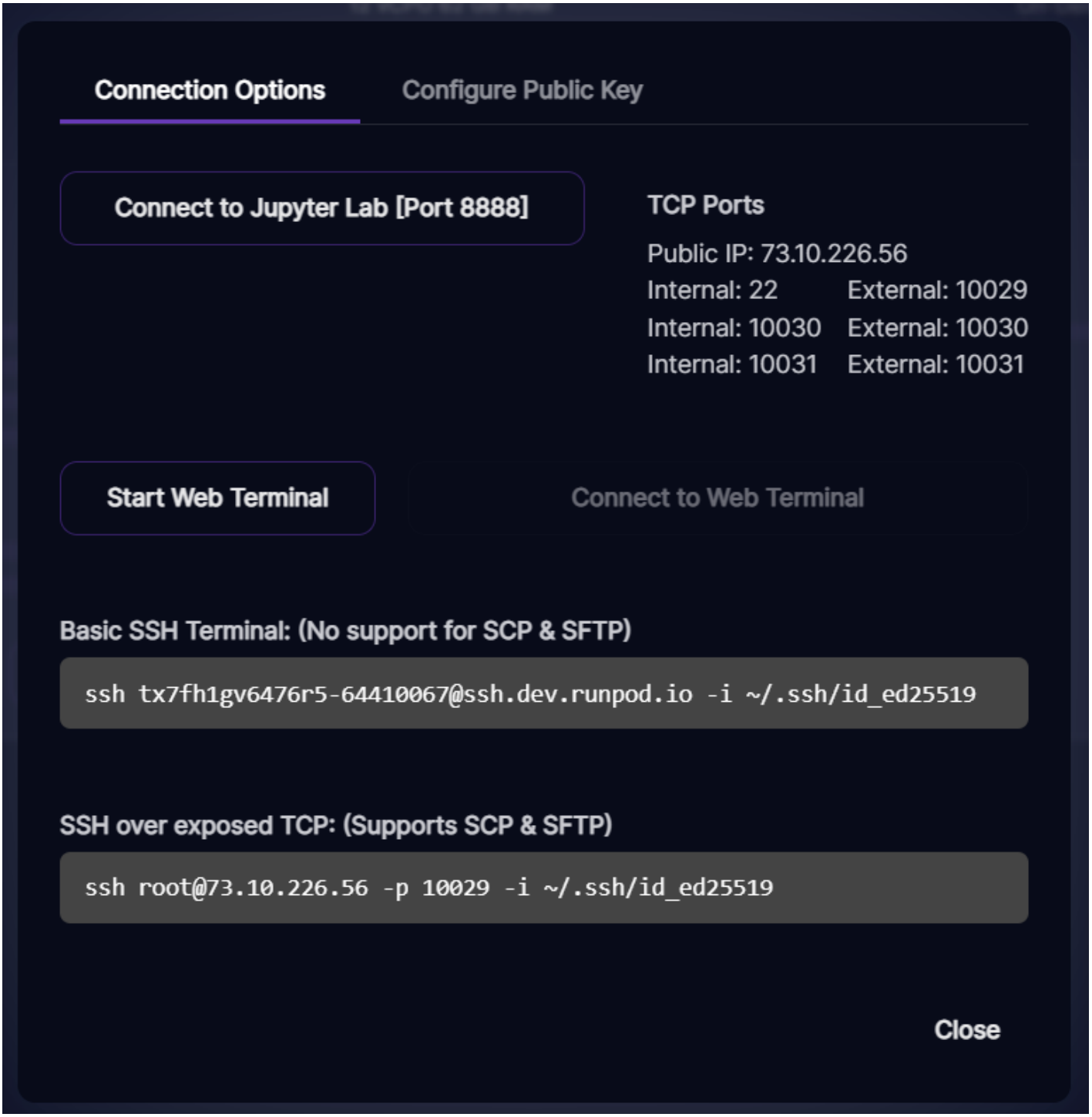
Be aware that the public IP could potentially change when using Community Cloud, but should not change when using Secure Cloud. The port will change if your pod gets reset.

Requesting a Symmetrical Port Mapping

For some applications, asymmetrical port mappings are not ideal. In the above case, we have external port 10027 mapping to internal port 4000. If you need to have a symmetrical port mapping, you can request them by putting in ports above 70000 in your TCP port field.



Of course, 70000 isn't a valid port number, but what this does is it tells RunPod that you don't care what the actual port number is on launch, but to rather give you a symmetrical mapping. You can inspect the actual mapping via your connect menu:



In this case, I have requested two symmetrical ports and they ended up being 10030:10030 and 10031:10031. If you need programmatic access to these in your pod, you can access them via environment variable:

```
RUNPOD_TCP_PORT_70001=10031
RUNPOD_TCP_PORT_70000=10030
```

Edit this page